Universität Zürich[UZH]

Institut für Informatik

Martin Glinz

Software-Qualität – Ausgewählte Kapitel

Kapitel 11

Messung und Prognose von interner Software-Qualität

Grundidee

Aus großen Datenbeständen über Software ...

mit geeigneten Analyseverfahren Informationen gewinnen, welche

- Aufschlüsse über die momentane interne Qualität geben
- Prognosen über qualitätsrelevante Phänomene erlauben

Datenbestände

- Versionsgeschichte von Software-Artefakten (insbesondere Code)
- Änderungsgeschichte
- Problemberichts-Datenbank
- Test-Suiten und -Protokolle
- Reviewberichte
- Projektdatenbank (Aufwände, Durchlaufzeiten, etc.)
- E-Mail-Verkehr / Chats der Entwickler
- ...

Klassische Artefaktanalyse

- Einfache Messungen

  Beispielsweise Erkennung überlanger Methoden oder Module

- Statische/Dynamische Analyse von Programmen

  Beispielsweise Erkennung von
  - nicht initialisierten Variablen
  - totem Code

- Analyse von Architekturen
  - Beispielsweise Erkennung von Anomalien in der Aufrufhierarchie

Analyse mit Data Mining

- Erkennung von Mustern und Anomalien

  Zum Beispiel: Die Testprotokolle eines Systems zeigen gehäuft Fehler bei der Benutzung einer bestimmten Bibliothek

- Lernen von Mustern (mit Verfahren des maschinellen Lernens)

  Zum Beispiel: Aus der Änderungsgeschichte wird gelernt, dass Änderungen in Modul X in den meisten Fällen Änderungen in den Modulen X1, A, F und R nach sich ziehen

Prognose qualitätsrelevanter Phänomene

Beispielsweise:

- Fehleranfälligkeit / Testintensität

  Zum Beispiel: Durch Maschinelles Lernen wurde eine signifikante Korrelation zwischen messbaren Größen im Versionsarchiv und der Fehleranfälligkeit eines Moduls erkannt → Daraus lässt sich ein Prädiktor für Fehleranfälligkeit ableiten

- Fehlervermeidung bei Änderungen

  Zum Beispiel: Prognose mitzuändernder Module, wenn ein Modul geändert wird

- Gelernte Korrelationen genügen – Kausalität ist nicht erforderlich

Vertiefung dieses Themas im Selbststudium

Lesen Sie die folgenden Papiere:

- Diehl, Zeller, Zimmermann (2006): Was Software-Archive erzählen
- Zimmermann et al. (2004): Mining Version Histories to Guide Software Changes
- Bird et al. (2009): Does Distributed Development Affect Software Quality?: An Empirical Case Study of Windows Vista.

Als ergänzende Lektüre:

- Sliwerski, Zimmermann, Zeller (2005).When do Changes Induce Fixes? On Fridays.
- Nagappan, Ball, Zeller (2006): Mining Metrics to Predict Component Failures.

Literatur

C. Bird, N. Nagappan, P. Devanbu, H. Gall, B. Murphy (2009). Does Distributed Development Affect Software Quality?: An Empirical Case Study of Windows Vista. *Communications of the ACM* **52**, 8. 85-93.

S. Diehl, A. Zeller, T. Zimmermann (2006). Was Software-Archive erzählen. In *Software Engineering 2006. Fachtagung des GI-Fachbereichs Softwaretechnik (SE 2006)*, Leipzig, März 2006. 39-50.

N. Nagappan, T. Ball, A. Zeller (2006). Mining Metrics to Predict Component Failures. *Proceedings of the 28th international conference on Software engineering (ICSE 2006)*. 452-461.

J. Sliwerski, T. Zimmermann, A. Zeller (2005).When do Changes Induce Fixes? On Fridays. *Proc. International Workshop on Mining Software Repositories (MSR)*, St. Louis, Missouri, USA, May 2005.

T. Zimmermann, P. Weißgerber, S. Diehl, A. Zeller. (2004). Mining Version Histories to Guide Software Changes. *Proceedings of the 26th International Conference on Software Engineering (ICSE 2004)*, Edinburgh, UK. 563-572.

Literatur – 2

T. Zimmermann, P. Weißgerber, S. Diehl, A. Zeller. Mining Version Histories to Guide Software Changes (2005). *IEEE Transactions on Software Engineering* **31**, 6 (June 2005). 429-445